Université Paris Diderot – Paris 7 **Programmation réseau**
L3 Informatique **Année 2011-2012, 2ème semestre**

TP n°5

Protocole SNTP

Introduction

Le protocole NTP (Network Time Protocol) permet de régler avec une grande précision l'horloge d'un hôte à partir d'une ou plusieurs sources de temps distantes. NTP est défini dans le RFC 1305, un document de 120 pages qui inclut une discussion détaillée des algorithmes de filtrage que les clients NTP doivent employer.

Pour beaucoup d'applications, la complexité de NTP n'est pas nécessaire. RFC 2030 définit SNTP (Simple Network Time Protocol), une version simplifiée de NTP qui ne définit que le protocole et pas les algorithmes à employer. Tout client ou serveur NTP est *a fortiori* un client ou serveur SNTP, mais l'inverse n'est pas vrai.

Le but de ce TP est d'implémenter un client SNTP.

1 Le protocole

Lorsqu'il détermine que c'est nécessaire, le client SNTP envoie au serveur une requête sous forme d'un paquet UDP qui contient la date à laquelle ce message est transmis. Le serveur répond (aussi vite que possible) avec une réponse qui contient quatre dates :

- la date à laquelle la requête a été transmise (selon le client) ;
- la data à laquelle la requête a été reçue (selon le serveur) ;
- la date à laquelle la réponse a été transmise (selon le serveur) ;
- la date à laquelle l'horloge du serveur a été réglée pour la dernière fois.

En outre, la réponse contient un certain nombre de données que nous ne considérerons pas dans ce TP, et qui permettent à NTP (mais pas nécessairement SNTP) d'avoir une idée de la précision des résultats.

On remarquera que la réponse contient la date que le client avait incluse dans la requête ; ceci permet au client d'identifier une réponse comme allant de pair avec une requête donnée, et donc de contourner les problèmes dûs à la nature non-fiable du transport.

1.1 Format des données

Date NTP Une date NTP est représentée sous la forme d'un entier de 64 bits exprimant un temps, en unités de 2^{-32} secondes, écoulé depuis 0 h le 1er janvier 1900. RFC 2030 définit le format d'une telle date comme suit :

0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31
Seconds
Seconds Fraction (0-padded)

Message NTP Les requêtes et les réponses NTP ont le format suivant :

0 1	2 3 4	5 6 7	8 9 10 11 12 13 14 15	16 17 18 19 20 21 22 23	24 25 26 27 28 29 30 31
LI	VN	Mode	Stratum	Poll	Precision
Root Delay					
Root Dispersion					
Reference Identifier					
Reference Timestamp (64)					
Originate Timestamp (64)					
Receive Timestamp (64)					
Transmit Timestamp (64)					

Les champs d'un tel message sont définis comme suit :

- *LI* sera ignoré dans ce TP, et vaudra toujours 0 ;
- *VN* est la version du protocole employée, et vaudra toujours 3 ;
- *Mode* est le type de message, et vaudra 3 pour une requête, 4 pour une réponse ;
- *Stratum*, *Poll* et *Precision* seront ignorés dans ce TP ;
- *Reference Timestamp* sera ignoré dans ce TP ;
- *Originate Timestamp* vaut 0 pour une requête, et, pour une réponse, est la copie du *Transmit Timestamp* de la requête correspondante ;
- *Receive Timestamp* vaut 0 pour une requête, et pour une réponse, est la date de réception de la requête correspondate ;
- *Transmit Timestamp* est la date de transmission de ce paquet.

Le paquet peut être suivi de 160 octets de données supplémentaires, que nous ne transmettrons jamais, et qui seront ignorés lors de la réception.

2 Code fourni

Vous trouverez sur la page web des TP un fichier NtpMessage.java pouvant vous aider. Il permet de manipuler un message NTP et récupérer facilement les champs du message NTP. Par exemple :

- `NtpMessage msg = new NtpMessage(packet.getData());`
  Renvoie un objet NtpMessage à partir d'un packet DatagramPacket.
- `NtpMessage.encodeTimestamp(array, pointer, timestamp)` Encode le *timestamp* dans le message *array* à partir de la position (en octet) *pointer*.
- `msg.toString()`
  Renvoie en chaîne de caractère le contenu du message.
- `msg.receiveTimestamp()`
  Renvoie te temps de réception du message (Receive Timestamp).

3 Exercices

Exercice 1 [Premier client NTP]

Écrivez en java un programme qui exécute les actions suivantes :

1. crée une requête NTP ;
2. envoie cette requête dans un paquet UDP vers une machine munie d'un serveur NTP ;
3. attend une réponse sur le port UDP ;
4. affiche le contenu de la réponse ;
5. calcule le décalage de l'horloge entre vous et le serveur ; (prendre en compte le délai de transmission)

Pour des raisons techniques, vous ne pouvez pas tester votre programme sur Internet, mais uniquement dans l'UFR.

Exercice 2 [Découverte de serveur]

Certains supports physiques, comme le réseau Ethernet utilisé dans les salles machines de l'UFR, permettent de diffuser une trame à tous les ordinateurs qui y sont connectés. Cette possibilité est re-exportée par la couche IPv4 à l'aide d'une adresse IP spéciale, appelée adresse de diffusion, ou adresse de *broadcast*. Ce mécanisme a été remplacé par le mécanisme plus général d'adresse *multicast*. Un paquet émis à destination de cette adresse sera reçu par tous les ordinateurs connectés sur le même lien physique. Ce paquet ne sera jamais retransmis par un routeur.

En envoyant une requête NTP à l'adresse *broadcast* de l'UFR (192.168.70.255), tous les serveurs NTP accessibles directement vous répondront.

Modifiez votre programme pour qu'il soit capable d'attendre plusieurs réponses. Vous pourrez supposer qu'après 3000ms de silence, il n'y aura plus de réponse.

1. Afficher les adresses des serveurs NTP ayant repondu à votre requête.
2. Déterminez le nombre de serveurs NTP de l'UFR.